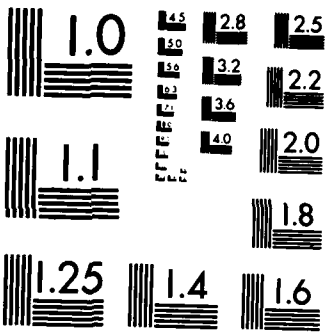
AD-A125 650 ALTERNATIVES TO OVERCOME THE COMMUNICATIONS PROBLEM OF FORMAL REQUIREMENT..(U) MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE R T MITTERMEIR ET AL. AUG 82 1/1

UNCLASSIFIED AFOSR-TR-83-0043 F49620-80-C-0001 F/G 9/2 NL



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(2)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFOSR-TR- 83-0043	2. GOVT ACCESSION NO.: A125650
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): ALTERNATIVES TO OVERCOME THE COMMUNICATIONS PROBLEM OF FORMAL REQUIREMENTS ANALYSIS	5. TYPE OF REPORT & PERIOD COVERED: TECHNICAL
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Roland T. Mittermeir, Pei Hsia, and Raymond T. Yeh	8. CONTRACT OR GRANT NUMBER(s): F49620-80-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Computer Science, University of Maryland, College Park MD 20742	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS: Directorate of Mathematical & Information Sciences, Air Force Office of Scientific Research, Bolling AFB DC 20332	12. REPORT DATE: August 1982
	13. NUMBER OF PAGES: 7
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Dr. Mittermeir is with the Inst. f. Angew. Informatik, Technische Univ. Wien, A-1040, Vienna, Austria; Dr. Hsia is with the Department of Computer Science & Engineering, University of Texas, Arlington TX 76010.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Even in spite of formal requirements analysis, problems persist in the full acceptance of software systems. These problems are basically due to communications problems between users and system experts.

This paper discusses some alternative approaches to overcome these problems. It weights their relative advantages and shortcomings against each other. The approaches discussed are: Objectives Analysis, Prototyping and the Scenario Technique.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ALTERNATIVES TO OVERCOME THE COMMUNICATIONS PROBLEM OF FORMAL REQUIREMENTS ANALYSIS

Roland T. MITTERMEIR
Inst. f. Angew. Informatik
Technische Univ. Wien
A-1040 Vienna AUSTRIA

Pei HSIA
Dept. of Comp. Sci. & Eng.
Univ. of Texas at Arlington
Arlington, TX 76010

Raymond T. YEH
Dept. of Comp. Science
University of Maryland
College Park, MD 20742

ABSTRACT:

Even in spite of formal requirements analysis, problems persist in the full acceptance of software systems. These problems are basically due to communications problems between users and system experts.

This paper discusses some alternative approaches to overcome these problems. It weighs their relative advantages and shortcomings against each other. The approaches discussed are: Objectives Analysis, Prototyping and the Scenario Technique.

1. THE PROBLEM

In recent years, it became well established in the computer science literature that projects should not start out with a rush into system design or implementation, but that they should be preceded by a thoroughly conducted requirements analysis phase.

The main arguments for conducting an in-depth requirements analysis before starting out with designing and constructing a system are mainly related to the problems of capturing as precisely as possible the users' (customers') intentions early enough. Only if requirements are laid out in enough detail can a thorough cost and time scheduling be conducted. If misunderstood requirements are detected during the requirements analysis phase, they can be corrected at little expense. If they are not detected during the requirements study, chances are very high that they will not be detected before testing within the customer environment starts, or, at least, before major portions of the new system have already been developed. The cost involved to accommodate changes this late is obviously several magnitudes higher than the cost of changing the requirements model in its pre-specification stage [1].

But, while conducting a thorough requirements analysis certainly helps to build firm foundations for a system development effort, the details about how to best do requirements analysis are still under dispute, and much of the literature is concerned rather with notational than with methodological aspects.

But, while a good notation is a necessary prerequisite for efficient communication, we must not forget that the best notation is of little help if one of the communicating partners is illiterate in it. Unfortunately, this illiteracy of requirements notations applies to most end users, irrespective of whether they are used to working in a technical or a commercial field and irrespective of their seniority.

Furthermore, there are even some deeper problems in communication among users, buyers (decision-makers) and analysts than just lack of a common notation or language. These problems are due to the inability of humans to see beyond their current frame of reference and, in general, lack of foresight. To adjust the frame of reference of users during requirements analysis, when they already focus on a particular technical solution and the analysis is down to the level of data, processes, quantities and real-time constraints, is hard, if not impossible. Both users and analysts are very soon involved in too much technical detail to recognize that they are traveling very well on a nice road, but the road may lead in a wrong direction. Users cannot discover this mistake, because the symbols that are shown on the analysts roadmap do not sufficiently relate to them and the analyst cannot see it either, because he lacks knowledge about the detailed environment.

To overcome these problems, two major possibilities exist. One is to stay on an abstract level but leave the area of day-to-day operations and activities. Proceeding from a birdseye's view will show a much broader range of alternatives that might be fruitfully investigated. Also, one will enjoy a much better look into the future from the higher level vantage point, than from the down to earth actual operations. From the higher level, a decision-maker might see that the objectives which seem to be extremely worthwhile to strive for today may be outdated or meaningless tomorrow, because simply his environment may be different then. The approach of investigating such pre-requirements problems is called objectives analysis [2].

An extremely contrary position is taken by the advocates of prototyping. Their point is that the best way to solve the communications problem is not to communicate at all, but rather give the

user a system to play with. If he likes it, it's fine; if not, he can at least say specifically what he did not like and this aspect will then be changed. After enough iterations, the system will stabilize, or one may even take the rather extreme point of view that due to a constantly changing environment, there will be no stabilization and the prototyping cycle will continue as long as the system exists [3].

A new proposition that tries to combine some favorable properties of both of the approaches mentioned above is the scenario technique [4]. There, the analyst tries to provide his partners with a set of concrete views of his future environment or his future system. Though some differences exist, the scenario technique can be fruitfully applied to support objectives analysis, as well as pure requirements analysis [5].

In the following chapters, we will discuss in more detail the basic concepts of objectives analysis, prototyping and the scenario technique. Then, we will try to make an assessment of each of these techniques and investigate the extent to which they are complementary or contradictory.

2. PRESENTATION OF THE INDIVIDUAL METHODS

We will present these three approaches to overcome the communications problem in alphabetical order. This allows us to begin with the method to be applied the earliest in the system life cycle, objectives analysis. Next we will look at the extremely antithetic approach of pure prototyping, and, finally, to fill the gap between the two approaches mentioned beforehand, we will discuss the scenario technique.

2.1 Objectives Analysis

Objectives Analysis, by itself, is a distinct phase of the system development cycle. It should be the first phase, conducted even before requirements analysis is started. The reason for conducting objectives analysis is that if one starts out from a given needs statement directly into requirements analysis for the system needed, it cannot be assured that one really can hit the optimal solution. This is so, because at the requirements level, a number of basic decisions are already presupposed. These decisions might have been taken on the basis of misconceptions or incorrect assumptions. Furthermore, it is not at all clear, which people in an organization are really behind those decisions. In order to have a system operating properly, however, it needs to be backed by a consensus of a large majority of its users.

Asking users about their own objectives will make them feel at home with whatever system comes up and will thus reduce potential frustrations and rejections due to the general immobility of people. In addition to these psychological advantages, objectives analysis will increase the planning horizon of the people that want a new system. This is because objectives are much more stable than the everyday activities. Therefore, even when the immediate operating environment of an application changes, it will be much easier for the system to accommodate to these changes if the premise for the system development has been long range objectives.

Another advantage of objectives analysis is that due to its abstraction from detailed activities, it allows to search for solutions to a problem even outside the immediate domain in which this problem occurred. For example at the requirements level, a solution has to be sought for any problem that comes up. On the objectives level, one can determine that the problem is not really in an area of central interest, and, therefore, it may be considered unimportant, or if the current situation is impossible to keep, a solution might be to drop the whole area in which the problem occurred.

The objective analysis phase consists of the following stages:

1. Context Analysis: In this stage the preliminary scope of the development effort is determined. To do this, the needs statement originally given and its justification are suitable because it is used only to identify the part of the organization that will be affected by the development effort. It helps the analyst to gain some basic understanding of the system and its immediate environment. In other methodologies, similar analyses are subsumed in an investigation phase [6] or "Ist-Analyse" [7].

During this stage, the analyst should learn about the basic functions that are to be performed by the target system and by the environment into which it will be embedded (e.g., the accounting department within company X, or car manufacturer Y within the national economy with special emphasis on the car market, steel market, car industry, etc.). The analyst must also identify the boundaries between the target system and its environment, and the major quantitative characteristics of it and of its environment (number of employees, number of customers, share of market, number of items produced or sold, etc.).

When all this information is collected, the persons that are affected by the development effort can be identified. These include the users of the new system, their managers, supervisors, and other personnel who benefit (or suffer) from the results of the target system. The list of persons thus obtained will be used as the basis for selecting the informants to be questioned during the following stage.

2. Objectives and Problem Analysis: This is the central stage in objectives analysis. Quite often, problems will point the analyst to the underlying objectives. A problem can be defined as the difference between an originally set goal and the actual goal attainment on a certain objectives dimension. For this reason, the problems expressed may be good indicators of objective dimensions. However, when solving a problem, it is equally important not to forget about those aspects of the organization's objectives which are well enough

attained so that they would not be mentioned in connection with problems. Since the ultimate solution must continue to accommodate these aspects, one cannot obtain a complete picture of the organization, using a mere problem analysis, but must rather find the major objectives that lead the activities of the organization.

The obtained objective dimensions need to be structured into goal - subgoal relationships, and the analysts must further determine whether there are complementary or adversary relationships between various goals and/or subgoals.

Studying just the organization's own objectives is not enough. The analysts need also to identify the objectives of adverse groups (e.g. competitors) which have the possibility to hinder full achievement of one's objectives. These "negative" objectives of groups outside of the organization will serve as restrictions in the later analysis and during the selection of the best solution.

Having obtained the objectives structure, the importance of its various dimensions, and the magnitude of inter-goal dependencies, the level of current attainment versus the desired attainment must be studied (parameterized, if possible, according to time and location).

It follows the integration of objectives structures of the different groups within the customer organization. This is a very difficult task. It involves usually a complex process during which the different parties within the organization will learn about each other's objectives and try to negotiate and/or arbitrate any differences. This negotiation process may be long and tedious. It is also important not to forget to verify the results obtained at each stage.

Finally, the objectives structure of the organization, as far as it is relevant to the current development effort, should be formally agreed and accepted.

3. Definition of the scope of the development effort: During this stage, the following are defined: the boundary between the system and its environment, the problem areas that need to be addressed, and the constraints (resources and resource limitations) under which the solutions need to be sought. These definitions should be verified and formally accepted.

To start objectives analysis, the following should be given:
- needs statements
- needs justifications
- the authorization to start the analysis.

As a result one should obtain:
- a formally accepted objectives structure
- a list of problems
- the scope of the development (analysis) area and the restrictions for the new solution
- some basic understanding of the organization (kind of enterprise, kind of products or services, macrostructure and history of the organization, management policies) and its environment (associated industrial world, local laws, cultural background, state of the associated markets, share of market).
- main tasks of the system
- available resources

In addition to these major results from objectives analysis, usually a set of byproducts which are most useful for determining which of the different potential solutions should be pursued, is obtained:
- a list of potential informants for the requirements analysis phase
- usually a set of alternative solutions to be considered.

A detailed discussion about the methodological aspects of objectives analysis, such as, how and from whom to get the required information; how an objectives structure can be documented best; and the kinds of verification and evaluation procedures that can be applied to a given objectives structure, as well as a detailed example of an objectives analysis, are given in [2]. The transition from objectives analysis to requirements analysis is explained in [8].

2.2 Prototyping

Objectives analysis is another phase on top of the development life cycle. In the opinion of some authors, this standard development cycle is already far too long and too clumsy, even without objectives analysis (e.g. [9]). These authors advocate replacing the paperwork involved with requirements- (and objectives-) analysis by a quickly generated working system, a prototype. But prototyping has received attention only recently in the software engineering field [10]. Therefore, it does not have a generally agreed upon definition yet.

While some authorities argue that the word "prototype" should be used in exactly the same way as it is used in any other engineering discipline. There, the first complete and fully operating product is called a prototype, whereas anything that has been built but is not intended to contain the full features of the working system, is rather called a model.

Other software engineers adopt a different terminology. They base their arguments on the difference between an industrial production process for hardware or physical equipment in contrast to the production process for the immaterial software. According to their viewpoint, a prototype would be a partly working system that may lack several aspects, such as infrequently used functions or capabilities, performance, reliability, facilities for ease of use, and so forth. Unfortunately, this is too vague a notion to serve as a definition. First, one has to know the purpose that will be pursued by this kind of "incomplete prototype" and only afterwards one can decide whether it should really carry the label "prototype".

Two broad classes of reasons exist as to why people want to develop a prototype. One is to reduce the risk of the system developer. He may want to have a prototype to study whether the task to pursue is feasible at all at the current state-of-the-art. Prototypes of this kind might usually be poor regarding any human engineering aspects. They may sometimes lack in functionality and provide solutions only to those problems that are the hardest to come by. In some cases, they may also fall short in performance aspects. In other cases, it may be just the performance area that is the reason why a prototype should be built at all. Without ignoring the feasibility problem, which is a problem that has to be addressed right at the boundary between requirements analysis and design, one might rather place prototyping tools for reducing the technological risk into the category of design tools than analysis support tools.

The other reason why one may develop a prototype is to reduce the application risk. In this case, obviously a good user interface or system interface will be important. Other aspects, such as functionality and performance, may be important or not, depending on the particular application area. But it is more than just the user interface that should be tested by prototypes of this kind. Rather, it is whether and how the whole system will fit into the application environment. To reach this goal, users must be sufficiently trained and the prototype has to operate in the real environment long enough, so that the problems of just getting started are overcome and the results of the experiment have the required validity. Tools to support this second kind of prototype application should definitely be considered as requirements tools.

On the basis of the above arguments, we will give the following definition for prototype, so that we have a firm basis for the arguments presented later in the paper:

"With prototype, we mean here an operating version of a (sub-) system that has all necessary characteristics of the final product. It differs from the final product only in the sense that both the developer and the user understand that there is no commitment on the part of the system developer with regard to the qualitative aspects of this (sub-) system prototype."

This definition is very rigid, but it allows us to clearly distinguish between prototypes and mock-up models (like faked user interfaces), or simulators, that execute parts of the final product to test their performance or robustness. On the other hand, the above definition is rather broad, since it still allows to call, e.g., the first complete version of a user interface a prototype. However, this would then be a prototype of the user interface only and not a prototype of the full system. The above definition is also very extensive in the sense that it nowhere specifies that a prototype has to be the first or one of the first versions of a system. We consider this to be important, because after an evolving system has undergone a major change, both the developer and the user might treat this system with care for a while and will not fully trust its perfect operation.

2.3 Scenarios

Scenarios assume an intermediate position among the abstract objectives structures discussed in section 2.1 and the down to earth prototypes mentioned above. They are similar to prototypes, because their aim is to provide users with an actual feeling about how their future system will look, operate or perform; but they will not be suitable to do any useful work. With objectives analysis, the scenario technique has the common characteristic of definitely showing the user projections into the future that are more than just one step ahead of time.

The scenario technique can be applied to support both the objectives analysis phase, as well as the requirements analysis phase of a project. During objectives analysis, scenarios can be used to relate to the people that strive for a new system, how their future environment may change due to their own activities or due to factors outside their control, and how they can react to these changes. The scenarios will therefore be on a very high level of aggregation and will extend in their scope well beyond the system that should eventually be developed. Another aspect where scenarios are helpful during objectives analysis is to show how achievement of conflicting objectives will affect other objectives of the people in the user organization. If the consequences of certain goal pursuing strategies can be made visible, it will be much easier for the analysis partners to agree on a final and common objectives structure. Furthermore, objectives related scenarios will help to identify hidden objectives. In this respect, their capacity is somehow comparable to the Rorschach test used by psychoanalysts [11].

Obviously, the media used for showing such scenarios have no relationship whatsoever to an actual prototype. Closest to the prototype would be the use of existing "advanced" societies or organizations that could be taken as potential examples of what will happen within the organization that just heads towards a new system development. For projections which reach deeper into the future, one has rather to rely on one of the more conventional techniques of presentation, be it written, oral, or audi-visual, with the help of films or video tapes especially created for this purpose.

During requirements analysis, scenarios can again be used to stimulate the user to think in terms of his future working environment and the opportunities and problems he will then have, instead of having him simply ask for those things to be solved that he particularly dislikes with his current support system. In addition to this function of adjusting the users' frame of reference, scenarios can be used to get some users out of the lethargy they might be in, because for years they never had a chance to reflect on their own working environment. If they are in such a situation, how can you expect them to speak up during the analysis and formulate their requirements for the next 5 or 10 years without being prompted by some

alternatives to their current world?

Further properties and merits of the scenario technique can be found in [5].

3. MERITS AND SHORTCOMINGS OF THE 3 APPROACHES

3.1 Objectives Analysis

The basic merit of objectives analysis is that it works like a rocket. It boosts the participants of the analysis for up, away from their day-to-day work. But the rocket analogy also shows its inherent problems if it cannot be supported by any of the other two techniques.

Objectives analysis is ideal in the sense that it is the only one of the three techniques that really make the people in the user organization reflect about what they are doing and what they are doing it for. This will definitely open the eyes of the participants and get them away from thinking in their current trails only.

On the other hand, its major disadvantage is that it may be felt to be dangerous, especially by middle management. It is far easier for top management to endorse such a thorough investigation about the objectives of the individual groups within an organization, since they are willing to conceive of this as part of their duty to lead an organization successfully into the future. Therefore, authorizing objectives analysis is for them just executing part of their power to control the way their organization should be run. Furthermore, top management will obtain, from the results of objectives analysis, a new picture of their organization, which by itself may turn out to be even more valuable than the development effort which will be started thereafter.

For middle management, however, the results of objectives analysis may turn out to be dangerous if it has not been conducted with great care. Changes of direction resulting from this analysis can be considered by them as a critique on their past performance. In addition to that, middle management is usually much more concerned than top managers about keeping their little kingdom socially stable. Objectives analysis will jeopardize this stability.

In addition to these very important socio-psychological problems, some purely technical problems exist. If the analysts or consultants are aware of them, they are easy to master; otherwise, they may endanger the whole project. One of these problems is abstraction, the other is time.

The abstraction problem is due to the fact that people on different organizational levels have a different frame of reference and a different planning horizon. Therefore, what may be a valid and well understood objective at the level of the board of directors may be rather obscure at the level of department heads within a division and totally out of reach for some people down at the shopfloor. But these differences can be easily solved in a structuring and translation process that goes from very high corporate level objectives down to task related goals. After all, it is the purpose of objectives analysis to come from these very high level objectives finally down to the tasks that should be performed in order to reach those objectives. If the analysis arrives at the task level, objectives analysis is completed. What starts then, is a negotiation process in which different opinions are sorted out and finally arbitrated. After this process of negotiations and learning about one's colleagues objectives, the project can progress to the selection of the best system alternative to be pursued and start with requirements analysis for this alternative.

The other critical aspect of objectives analysis is time. One can rather take it as a rule that there will be conflicting objectives among the different groups within an organization. To solve these conflicts will take time. If either the analysts or certain groups within the organization try to rush too fast through this process of negotiations, there is no chance that a process of learning and understanding each other can get started. The defeated party will finally boycott the solution reached on the basis of objectives analysis to at least the same extent as if they would have never been asked their opinion.

On the other hand, moving too slowly is also dangerous, since during objectives analysis, nothing tangible is produced. Notably those participants that have been most active during the initial phases may become disappointed if their good suggestions are not implemented immediately. Skillful implementation of some changes and proper advertising those changes throughout the organization will help to solve this problem.

3.2 Prototyping

Prototyping can never run into the problem just mentioned above. Overstressing the point, with prototyping, the new system comes even before somebody asked for it. In addition to this outstanding advantage in timeliness, prototyping also has the advantage that it is always done at a concrete level. There cannot be any problems or confusion due to misunderstood terms or arguments put forth on the wrong level of abstraction.

Against this, we see the big problem of not leaving the old, worn-out tracks and also a problem with evolvability in the future.

These problems result on one hand from the fact that users are usually not presented with a series of prototypes. (Extremely large and costly projects might warrant to do so, but for a project that should be classified simply as "large," it seems safely to make the above statement.) Therefore, even while users might be encouraged to make suggestions about how the prototype they have at hand should be improved, these suggestions will usually be only reactive. It is hard to expect users to become creative enough to suggest an entirely new concept for solving their problems on the basis of their experience with a particular prototype only. Therefore, the action is back with

the system developers. The difference to the "start out with a good design"-days might be just that this design can be implemented much faster. Therefore, the implementation is less costly and it is not considered a catastrophe if users disapprove of what they are given.

While this point might be still acknowledged by hard prototyping advocates, the argument for lack of evolvability will be much more disputed. It will be disputed on the ground that the prototype could be implemented quickly. Therefore, if the system should respond to a change in the environment, whats easier than to implement very quickly another prototype.

Two arguments should be held against this simplistic notion. The first is that the problem may not be solved by simply installing another prototype. The problem may be rather that the currently running prototype has been installed in the first place. Getting an acceptable system without looking around for alternatives may turn out to be the reason why several years of (application) work have been invested into a direction off the actual trend. To come back to the main street of profitable activities may be much harder than simply to install another prototype.

The other argument is that we must not conceive of a system as consisting of its software portion only. Not only hardware, but also people, procedures and know-how should be considered when speaking about system changes. For this reason, it is of little help if it turns out that the software can be quickly redeveloped if a major turn is needed. Even higher investments than in the software lie in the corporate data source and in the personnel of the organization with its know-how and procedures. Notably with the data sources it is very often impossible to regain data that has been lost in previous time due to lack of foresight. It may seem trivial, but, in practical situations, an otherwise perfect system solution may turn out to become economically infeasible if, say, the birthdate or the social security number of over a million of customers is not readily available. For problems of this sort, another fast prototype cannot provide any help. Help comes only from foresight and getting a long range perspective. Here, objectives analysis definitely works best and prototyping will be the worst of the three techniques discussed.

It is also this operating environment which will make the introduction, use, and potential rejection of prototypes rather costly, no matter how cheap the development of the prototype itself ever may be.

3.3 Scenario Technique

Fortunately, with the scenario technique, we need not paint that much in black and white as we did in the previous sections. On one hand, if not applied adequately, it may share the deficit of prototyping in putting the user into a passive role. But this should not be the case and is against the intentions of the developers of this technique.

But even if misapplied, there is less danger to produce reactive users only than with full prototyping. Due to the limited expense of simply presenting another scenario, the general rule should be that users are presented with more than only one of them. If these alternative scenarios cover a broad enough spectrum of possibilities, one can even expect that users can overcome their own limitations by becoming creative sources of alternatives which would not have been conceived previously.

The possibility to provide users with very diverse pictures of what they are heading for puts scenarios also far ahead of prototypes with risking to stay in worn out tracks until it is too late to turn.

On the other hand, scenarios are not real. This makes them very cheaply to construct. There are basically no design costs involved and the development costs will be limited and in any case far less than those of a prototype. Therefore, they can serve as black box models only. They cannot answer any questions about the technology risk involved with a particular proposal; neither can they show how adequately a proposal that looks very good at a short presentation will perform during an extensive test period. If these are the hard questions to be addressed, one would be back in the realm of prototypes for at least the critical portions of a system, supported by simulation for those parts of the system that have not yet been completed.

With regard to objectives analysis we do not see any contradictions between the two techniques. In fact, we consider that scenarios will support objectives analysis in the same way as (other) scenarios will support requirements analysis. However, there is also one problem the scenario technique shares with objectives analysis. With the much more concrete scenarios this may be even worse than when working on the abstract level of objectives. It is the problem of getting the final solution fast enough. This is due to the fact that "the appetite comes with the eating". If users see what a nice world is ahead of them, it will be very hard for them to wait for the full duration of an orderly development cycle to finally get all these good things. And there closes the circle. After knowing exactly what should be done, little can be argued against reaching a quick and nice implementation by using prototyping tools and techniques.

4. CONCLUSION

Objectives analysis, prototyping and the scenario technique have been presented to enhance the current methods of formal requirements analysis. The scenario technique has been successfully used by P. Hsia in some of his projects. Objectives analysis, partly supported by scenarios (though the word "scenario" has not been used by him at this time) has been conducted in several projects in which R. Mittermeir has been involved. The experience gained in those projects is that ade-

quate timing is one of the key factors for success of objectives analysis, and that scenarios can indeed be used in an extremely positive way to reconcile different opinions, or to adjust (or at least highlight) the differences in the frame of reference of different organizational groups.

On the basis of this experience, we do recommend the use of these concepts to arrive at better systems as well as at better accepted systems. Use of prototyping is considered a heavier weapon. Therefore, we recomend to use it only when one can be sure that it will hit at least very close to the actual target.

ACKNOWLEDGEMENT

We all want to acknowledge the cooperation of the various organizations that provided us with the basis for a comparison of this sort. In addition to that, R. Mittermeir wants to acknowledge several good ideas that he obtained during lengthy discussions with members of the research project S-23, sponsored by the Austrian Research Promotion Fund.

This work was supported in part by U.S. Army Contract DASG 60-81-C-0050; also by the Air Force Office of Scientific Research Contract AFOSR-F49620-80-C-001

REFERENCES

[1] B. W. BOEHM: "Software engineering economics", Prentice-Hall Inc., Englewood Cliffs, N.J. 1981.

[2] R. T. MITTERMEIR, N. ROUSSOPOULOS and R. T. YEH: "Objectives Analysis - a phase of the system life cycle", University of Maryland, 1982. (submitted for publication)

[3] C. FLOYD: "A Process-Oriented Approach to Software Development", Proc. ICS '81, London, March 1981.

[4] J. W. HOOPER and P. HSIA: "Scenario based prototyping for requirements identification", to appear in: ACM Software Engineering Notes.

[5] P. HSIA and R. T. MITTERMEIR: "Conceptual foundations for a scenario-driven approach to assess system requirements", working paper, UM College Park and UT Arlington, 1982.

[6] I. MIYAMOTO and R. T. YEH: "A software requirements and definition methodology for business data processing", Proc. 1981 National Computer Conference, AFIPS Conference Proc., Vol. 50, pp. 571 - 581.

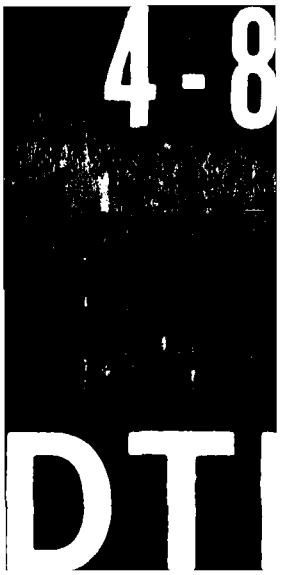
[7] H. WEDEKIND: "Systemanalyse", Carl Hanser Verlag, Muenchen, BRD, 1976.

[8] R. T. MITTERMEIR, I. MIYAMOTO, N. ROUSSOPOULOS and R. T. YEH: "Requirements Analysis - An integrated approach", TR-1155, University of Maryland at College Park, 1982.

[9] J. MARTIN: "Application development without programmers", Prentice-Hall Inc., Englewood Cliffs, N.J. 1982.

[10] M. ZELKOWITZ (ed.) Proceedings of the 2nd Software Engineering Symposium: "Rapid Prototyping", Columbia, MD, 1982.

[11] H. KERNER and L. WENINGER: "Vorgehensweise f. Systemplanung - Vorschlag f. ein Planungsmodell", Arbeitspapier der Gruppe Gestaltungsmethodik, Projekt S-23, 1982.


4-8
DT